Koncept · 23 maj 2026

← Föregående·Nästa →

De 24 orden: vad en kryptografisk identitet är

En kryptografisk identitet är inte ett lösenord: ingen server sparar den och den kan inte återställas. En didaktisk förklaring av BIP39-mekanismen, varför exakt tjugofyra ord, och vilken reell vikt som vilar på den som besitter dem.

För att förstå varandra: Om du glömmer ditt lösenord till Gmail återställer Google det åt dig. Om du tappar bort de 24 orden som utgör en kryptografisk identitet finns det ingen att be om dem. Det är inte så att proceduren är sträng – det är att det inte finns någon i andra änden. Den skillnaden är hela skillnaden.

Skillnaden mellan ett lösenord och en identitet

Ett lösenord i den klassiska internetmodellen är inte användarens identitet. Det is ett bevis. Användaren har en identitet – ett namn, en e-post, ett kundnummer – och för att bevisa för en server att man är den man utger sig för att vara, presenterar man ett lösenord som servern jämför med ett lagrat avtryck. Om avtrycken stämmer överens beviljar servern sessionen. Om lösenordet tappas bort förblir användaren samma användare; det man förlorar är beviset, och det finns en återställningsprocedur – ett e-postmeddelande till den registrerade adressen, en säkerhetsfråga – för att återställa det.

En kryptografisk identitet fungerar på ett annat sätt. Det är inte en legitimationsuppgift som någon jämför med ett lagrat avtryck; det *är* en komplett matematisk hemlighet i sig själv. Det spelar ingen roll var den befinner sig – på ett papper, i en enhet eller till och med på en främmande server: identiteten existerar genom sin matematik, inte genom vem som validerar den. Här dyker en egenskap upp som liknar den vi såg i «Vad SHA-256 egentligen är»: innehav bevisas inte genom att visa upp hemligheten, utan genom att använda den för att signera. Signaturen som produceras på detta sätt kan kontrolleras av vem som helst med ett offentligt värde som härleds matematiskt från själva hemligheten, utan behov av att känna till hemligheten själv och utan att en tredje part medlar i kontrollen. Den som har hemligheten är identiteten; den som förlorar den upphör att vara det. Domen är kategorisk: **det finns ingen att be om att få identiteten tillbaka. Den personen existerar inte, för hen hade den inte från början.**

Vad tjugofyra ord representerar

Den kryptografiska identiteten representeras vanligtvis av en matematisk hemlighet på trettiotvå bytes – tvåhundrafemtisex bits. Ett tal som är svårt att komma ihåg och ännu svårare att skriva av utan fel. Kryptobranschen löste detta problem 2013 med en liten och elegant standard kallad BIP39: ett sätt att representera dessa tvåhundrafemtisex bits som en sekvens av tjugofyra ord tagna från en officiell lista på tvåtusenfyrtioåtta. Aritmetiken bakom passar elegant; de som vill se den i detalj hittar den i marginalen.

Räkningen börjar från slutet. Vi vill representera de tvåhundrafemtisex bitarna i hemligheten genom att lägga till åtta bits kontrollsumma: totalt tvåhundrasextifyra bits. Om vi fördelar dem på tjugofyra ord – ett hanterbart antal för att anteckna och diktera utan förlust – måste varje ord bidra med exakt elva bits information. Och elva bits är två upphöjt till elva möjligheter, det vill säga tvåtusenfyrtioåtta. Därför har det officiella BIP39-vokabuläret exakt den storleken: listan existerar anpassad efter problemet, inte tvärtom.

Räkningen är inte dekorativ. Om någon skriver av tjugotre ord korrekt och gör ett fel i det tjugofjärde, kommer kontrollsumman att upptäcka det: programvaran kommer att säga "denna sekvens är inte giltig". Om någon skriver av alla tjugofyra korrekt kommer programvaran att härleda samma identitet utan tvetydighet. Valet av ordlistan är också medvetet: orden i BIP39-vokabuläret är korta, skiljer sig från varandra, utan diakritiska tecken, valda för att minimera fonetiska och ortografiska förväxlingar. Det är ett ordförråd utformat för att kommas ihåg, skrivas och dikteras av människor utan förlust.

Från fras till nyckel

De tjugofyra orden är inte den kryptografiska nyckeln som signerar meddelanden. De är en återställningsbar representation av den ursprungliga entropin som, genom en deterministisk process kallad PBKDF2, transformeras till ett seed på sextiofyra byte. Från detta seed härleds, också deterministiskt, de konkreta kryptografiska nycklar som användaren använder: en privat nyckel för att signera och en motsvarande publik nyckel som publiceras för att verifiera signaturerna. Samma mekanism i olika system: kryptovalutor använder secp256k1-kurvan; Signal-protokollet och många moderna system använder Ed25519 på Curve25519-kurvan. För en specifik kurva som Ed25519 tar standarderna BIP32 och SLIP-0010 detta seed på sextiofyra byte och härleder deterministiskt de trettiotvå byte som utgör den effektiva signeringsnyckeln — samma trettiotvå byte som kodexemplet i nästa avsnitt börjar med.

Detta är standardsättet som hela branschen presenterar mekanismen för användaren — kryptovaluta-plånböcker, hanterare av decentraliserad identitet, Signal i sin persistenta identitetsdel, Solo2 bland dem —: användaren ser i praktiken aldrig seedet eller de härledda nycklarna. Han ser de tjugofyra orden när han skapar sin identitet och skriver, valfritt, ner dem på ett papper. Orden reser sedan mellan hans enheter när han vill migrera identiteten: han skriver in dem i den nya applikationen, applikationen härleder samma seed, samma nycklar, samma identitet. Det är en bärbar, kryptografiskt solid och, inom rimliga gränser, memorerbar mekanism.

Hur man signerar med nyckeln (en penseldrag i Zig)

I Zig, när man väl har seedet på trettiotvå byte härlett från de tjugofyra orden, ryms signering av ett meddelande med Ed25519 på några få rader:

```
const std = @import("std");
const Ed25519 = std.crypto.sign.Ed25519;

// 'semilla' son los 32 bytes derivados de las 24 palabras.
const par = Ed25519.KeyPair.create(semilla);

// Firmar un mensaje con la clave privada:
const mensaje = "Este mensaje lo escribí yo.";
const firma = try par.sign(mensaje, null);

// Cualquiera con la clave pública del par puede verificar:
try Ed25519.Signature.verify(firma, mensaje, par.public_key);
```

Signeringsoperationen producerar sextiofyra byte —kallat en signatur— som bara kunde ha genererats från den motsvarande privata nyckeln. Verifieringen är publik: vem som helst med den publika nyckeln kan kontrollera att signaturen motsvarar meddelandet. Utan den privata nyckeln kan ingen producera en giltig signatur för det meddelandet; med den publika nyckeln kan alla upptäcka om en signatur är giltig. Denna asymmetri är vad som gör det möjligt för undertecknaren att bevisa författarskap utan att dela hemligheten.

Föregående exempel är manualens minimala version. I den verkliga Solo2-koden går kedjan genom två filer, en i JavaScript som lever i användarens webbläsare och rekonstruerar entropin från de tjugofyra orden, en annan i

Zig i biblioteket *zcatcrypto* som tar den entropin och härleder de konkreta kryptografiska nycklarna. Vi börjar på webbläsarsidan:

```
// solo2/web-app/js/lib/bip39.js
async function mnemonicToEntropy(mnemonic, lang) {
    const validation = await validateMnemonic(mnemonic, lang);
    if (!validation.valid) {
        return { entropy: null, valid: false, error: validation.error };
    }
    const wordlist = WORDLISTS[lang || 'en'];
    const words = mnemonic.trim().split(/\s+/);

    // Cada palabra aporta 11 bits (su índice en la lista de 2048).
    let bits = '';
    for (let i = 0; i < words.length; i++) {
        bits += wordlist.indexOf(words[i]).toString(2).padStart(11, '0');
    }

    // 24 palabras = 264 bits. Los primeros 256 son la entropía.
    const entropyBytes = new Uint8Array(32);
    for (let j = 0; j < 32; j++) {
        entropyBytes[j] = parseInt(bits.slice(j * 8, (j + 1) * 8), 2);
    }
    return { entropy: entropyBytes, valid: true };
}
```

Dessa trettiotvå byte entropi, tillsammans med ytterligare trettiotvå som härleds i samma steg, reser till Zigs WebAssembly-modul som genererar själva Ed25519-nycklarna. Den fullständiga funktionen, med sin slutliga minnesrensning, får plats på en skärm:

```
// zcatcrypto/wasm/bindings/identity.zig
const Ed25519 = std.crypto.sign.Ed25519;
const X25519  = std.crypto.dh.X25519;

export fn identity_generate() ?*IdentityHandle {
    var seed: [64]u8 = undefined;
    if (!common.getRandomBytes(&seed)) return null;

    const handle = common.wasm_allocator.create(IdentityHandle) catch return null;

    // Bytes 0..31: semilla determinista del par Ed25519 (firma).
    const sign_kp = Ed25519.KeyPair.generateDeterministic(seed[0..32].*) catch {
        common.wasm_allocator.destroy(handle);
        return null;
    };
    handle.sign_secret = sign_kp.secret_key.toBytes();
    handle.sign_public = sign_kp.public_key.toBytes();

    // Bytes 32..63: secreto X25519 (para acordar claves de cifrado con el otro).
    handle.exchange_secret = seed[32..64].*;
    handle.exchange_public = X25519.recoverPublicKey(handle.exchange_secret) catch {
        common.wasm_allocator.destroy(handle);
        return null;
    };
```

```
    @memset(&seed, 0);  // Borra la semilla de la memoria.
    return handle;
}
```

Två detaljer är värda att notera. För det första: samma frö (seed) producerar alltid samma nyckelpar — det är precis detta som gör det möjligt att återställa identiteten genom att ange de tjugofyra orden på en ny enhet. För det andra: fröet raderas explicit från minnet i den sista raden. Efter den punkten skulle inte ens funktionen själv kunna rekonstruera nycklarna; användarens ord skulle vara den enda källan.

För dem som vill kontrollera det med små tal. Signaturschemat kan genomgås i sin helhet med siffror som är tillräckligt små för att göra beräkningarna för hand. De som föredrar att inte gå in på aritmetik kan hoppa över detta block utan att tappa tråden i artikeln; de som vill se mekanismen fungera steg för steg hittar den här. **De offentliga reglerna**, som vem som helst kan läsa: ett primtal $p = 23$ (i verklig Ed25519 är det på cirka sjuttiosju siffror; vi använder tjugotre så att beräkningarna får plats på en sida), en bas $g = 2$ vars ordning i denna grupp är $q = 11$, och konventionen att all aritmetik med g görs *módulo p* och alla exponenter reduceras *módulo q*. **Det privata valet**, ett enda och aldrig delat: hemligheten $x = 6$. Det är identiteten.

Steg 1 — Den offentliga delen av identiteten. Den beräknas en gång och publiceras öppet.

$$y = g^x \bmod p$$

$$y = 2^6 \bmod 23 = 64 \bmod 23 = 18$$

Den offentliga delen av identiteten är **18**. Vem som helst kan ta den och använda den för att verifiera signaturer gjorda med denna identitet. Ingen kan, genom att bara observera 18, återställa hemligheten 6: det är det diskreta logaritmproblemet som vi kommer att återkomma till i slutet.

Steg 2 — Signera ett meddelande. Innehavaren av identiteten vill signera meddelandet $m = 7$. Han börjar med att välja ett nytt slumpmässigt värde $k = 4$, som kommer att användas endast en gång och aldrig delas (i verklig Ed25519 härleds k deterministiskt från meddelandet och hemligheten för att undvika faran med återanvändning, men rollen det spelar är precis denna). Sedan beräknar han tre tal:

$$r = g^k \bmod p = 2^4 \bmod 23 = 16$$

$$e = H(r, m) \bmod q = (16 + 7) \bmod 11 = 1$$

$$s = (k + x \cdot e) \bmod q = (4 + 6 \cdot 1) \bmod 11 = 10$$

Signaturen är paret **(r, s) = (16, 10)**. Den reser öppet tillsammans med meddelandet. Vem som helst kan läsa den. Didaktisk not: i verklig Ed25519 är funktionen H SHA-512, kryptografiskt robust; här använder vi förenklingen $e = (r + m) \bmod q$ så att läsaren kan genomgå stegen utan att behöva beräkna en hash. Algoritmens struktur är densamma.

Steg 3 — Verifiera signaturen. Verifieraren har den offentliga delen $y = 18$, meddelandet $m = 7$ och signaturen $(r, s) = (16, 10)$. Han rekonstruerar e på samma sätt — $e = (16 + 7) \bmod 11 = 1$ — och kontrollerar om denna likhet gäller:

$$g^s \bmod p \stackrel{?}{=} r \cdot y^e \bmod p$$

Beräknar de två sidorna separat:

Izquierda: $2^{10} \bmod 23 = 1024 \bmod 23 = 12$

Derecha: $16 \cdot 18^1 \bmod 23 = 288 \bmod 23 = 12$

De två sidorna ger **12**. Signaturen är giltig. Vem som helst med den offentliga delen 18 kan nå denna slutsats utan att någonsin ha vetat att hemligheten var 6.

Och en tredje part som försökte förfalska? Eva har sett allt offentligt passera genom kanalen: $p = 23$, $g = 2$, $q = 11$, $y = 18$, $m = 7$, $r = 16$, $s = 10$. För att signera ett *annat* meddelande i denna identitets namn skulle hon behöva känna till x. Hennes enda väg är att fråga sig själv: "för vilken exponent x gäller $2^x \bmod 23 = 18$?". Med $p = 23$ kan hon prova 0, 1, 2, 3, … och hitta det på några sekunder. Men genom att ersätta 23 med ett primtal av Ed25519:s verkliga dimensioner överstiger rymden av möjliga exponenter antalet atomer i det observerbara universumet. **Det finns idag ingen algoritm känd av mänskligheten som kan genomlöpa den rymden på mindre än miljarder år.** Det är samma diskreta logaritmproblem som ligger till grund för Diffie-Hellman i den föregående artikeln, tillämpat här på signaturschemat.

Det vi just har gått igenom är *exakt* Schnorr, signaturschemat som Ed25519 är en variant av, anpassad till en elliptisk kurva. I verklig Ed25519 görs alla operationer på punkterna på en specifik kurva (Curve25519) istället för på heltal modulo ett primtal, och funktionen H är SHA-512 istället för den leksakssumma vi använde ovan. De två ersättningarna är implementeringsjusteringar — att få kryptografisk resistens mot brute force, att få ytterligare säkerhetsegenskaper för k. Den algoritmiska strukturen, de tre operationerna och orsaken till asymmetrin är desamma.

Ett kort uppehåll är på sin plats här, eftersom hela kedjan vid en snabb blick kan förväxlas med en annan primitiv i trion: hashen. Det är det inte. En hash är en unik funktion som komprimerar — många byte går in, ett kort avtryck kommer ut, där slutar vägen. En kryptografisk identitet är ett komplementärt matematiskt par: hemligheten stannar kvar och signerar; dess offentliga motsvarighet publiceras och verifierar. Där hashen kollapsar information i en enda riktning, etablerar identiteten en asymmetri mellan två halvor. Hashen vittnar om vad som sades; identiteten vittnar om vem som sade det.

Vad frasen inte är

Tre vanliga missförstånd bör redas ut. Frasen är inte ett lösenord i egentlig mening: den jämförs inte med ett fingeravtryck lagrat på en server; den matas in i användarens enhet för att matematiskt rekonstruera identiteten. Frasen återställs inte: om den tappas bort finns det ingen att be om den; om den dupliceras dupliceras även identiteten. Frasen är inte en legitimation som kan skiljas från identiteten: frasen *är* identiteten. Den som har den kan agera som den identiteten, utan ytterligare tillstånd, utan auktoriseringsprocess, utan möjlighet till återställning.

Just denna tredje egenskap är vad som ändrar sakens tyngd. Ett förlorat lösenord är ett administrativt besvär. En förlorad kryptografisk identitet är identiteten själv. Ett papper med frasen som hittas av tredje part är inte en risk för kontostöld: det är överlämnandet av hela identiteten. Systemets löfte — att ingen kan återkalla din identitet eller blockera dig godtyckligt — åtföljs oskiljaktigt av ansvaret — att du är den enda väktaren av något som ingen kan återställa åt dig.

Löftet och tyngden

Modellen för kryptografisk identitet får ofta beteckningen *självsuverän* —self-sovereign i den angelsaxiska litteraturen—. Valet av ord är medvetet och beskriver tillståndet ganska exakt. Användaren är suverän över sin identitet i en nästan medeltida mening: den tilldelas inte av någon kung, någon utfärdare, någon central myndighet; ej heller kan den dras tillbaka av någon av de föregående. Men i likhet med den medeltida monarken bär användaren också hela konsekvensen av sina misstag: det finns ingen regent som fattar beslut i sitt ställe om han förlorar sigillet.

Valet mellan en identitet som hanteras av en tredje part och en självsuverän identitet har inget universellt korrekt svar. För ett oviktigt forumkonto är hanterad identitet troligen proportionell mot risken. För en professionell identitet som signerar juridiskt bindande dokument, för en ekonomisk identitet som bevakar egna besparingar,

för en professionell kommunikationsidentitet med klienter som har anförtrott känslig information, ändras frågan. Där upphör frågan att vara «är det bekvämt?» och blir «vem, förutom jag själv, har makten att agera som jag, och under vilka omständigheter?».

Var denna mekanism dyker upp i verkliga system

BIP39 föddes i Bitcoin-världen 2013 och spreds snabbt till hela kryptovaluta-ekosystemet: varje seriös plånbok accepterar idag en BIP39-fras på tolv eller tjugofyra ord som backup för innehavarens ekonomiska identitet. Utanför kryptovalutor uppträder samma underliggande koncept — ett kryptografiskt par som bevisar upphovsmannaskap utan en mellanhand — i andra system med annorlunda syntax. De SSH-nycklar som en systemadministratör använder för att få åtkomst till sina servrar är ett klassiskt fall: en privat nyckel som administratören sparar på sin maskin och en offentlig som kopieras till varje server; ingen enhet som kan jämföras med en centraliserad tjänst ingriper. Signal-protokollet använder Ed25519 med persistent nyckelmaterial på enheten; det europeiska eIDAS vilar, i sin del om kvalificerad signatur, på samma kryptografiska princip, med skillnaden att nyckeln förvaras av en kvalificerad betrodd tjänsteleverantör istället för användaren.

Solo2, utgivningsplattformen för denna publikation, använder en BIP39-fras på tjugofyra ord som identitet för varje användare. Användaren ser orden en gång när kontot skapas. De lagras inte på någon Solo2-server eller hos någon annan: om användaren antecknar dem och förvarar dem säkert behåller de sin identitet för alltid. Om de tappar bort dem så är de borta. Det är den logiska konsekvensen av en arkitektur utan en mellanliggande operatör: om Solo2 kunde ge tillbaka identiteten till användaren som förlorade den, skulle de också kunna ge den till vem som helst som sätter press på Solo2 för att få den.

För den professionella läsaren

Fyra överväganden för den som utvärderar att anta en kryptografisk självsuverän (autosoberana) identitet i ett professionellt sammanhang:

1. Frasen är identiteten. Fysisk förvaring — papper, flera kopior på olika platser, eventuellt ingraverad metall för långsiktig användning — erbjuder fler garantier än digital förvaring, som ökar attackytan utan att minska risken för förlust.
2. Det finns ingen återställning. Att utforma processen utifrån antagandet att den primära kopian en dag går förlorad är mycket klokare än att upptäcka det den dagen den går förlorad. En andra geografiskt åtskild kopia löser nästan alla scenarier.
3. Det är inte samma sak som ett eIDAS-kvalificerat certifikat. För kvalificerad signatur i Unionen — notariehandlingar, vissa ärenden hos förvaltningen — kräver lagstiftningen en kvalificerad leverantör som förvarar nyckeln. Kryptografisk självsuverän identitet tjänar för professionell kommunikation och dokumentunderskrift med bevisvärde, men ersätter inte automatiskt det kvalificerade certifikatet i de fall där regeln kräver det.
4. Om identiteten ska överföras — arv, professionell succession, verksamhetsavslut — bör man förbereda proceduren före, inte efter. Formella procedurer med kuvert förseglade med lack (lacre), instruktioner till en testamentsexekutor, deponering hos notarie, är klassiska arrangemang som är helt kompatibla med tillgångens kryptografiska natur.

Denna artikel avslutar den konceptuella trio som inledde cykeln — hash, kryptering, identitet —. De tre idéerna bygger på varandra: hash ger det oföränderliga fingeravtrycket, kryptering ger konfidentialitet utan en betrodd tredje part, identitet ger upphovsmannaskap utan en beviljande tredje part. De tre delar en egenskap som inte heller är ideologisk: de överför, från den som hanterar en tjänst till den som använder den, tekniska förmågor som traditionellt låg hos operatören. De överför också ansvar med dem. Att tala ärligt om någon av de tre kräver att man också talar om de andra två.

Källor och vidare läsning

- Palatinus, M.; Rusnak, P.; Voisine, A.; Bowe, S. — *BIP-0039: Mnemonic code for generating deterministic keys*, förslag till förbättring av Bitcoin från 2013. De facto-standard för återställningsfraser i kryptoindustrin.
- RFC 8032 — Edwards-Curve Digital Signature Algorithm (EdDSA), inklusive Ed25519. IETF, januari 2017. Normativ specifikation av det signaturschema som används i stora delar av den samtida industrin.
- RFC 2898 — PKCS #5: Password-Based Cryptography Specification, version 2.0. IETF, september 2000. Definierar PBKDF2-algoritmen som används i BIP39-härledning från fras till seed.
- Förordning (EU) 910/2014 (eIDAS) och dess utveckling genom förordning (EU) 2024/1183 (eIDAS 2) — europeiskt ramverk för elektronisk identitet och kvalificerad signatur. En annan ordning än den självsuveräna, men konceptuellt stödd av samma kryptografiska primitiver.
- Allen, C. — *The Path to Self-Sovereign Identity* (2016). Kanonisk text om principerna och åtagandena i den självsuveräna modellen, tidigare men relevant för förståelsen av familjen av samtida lösningar.

← FöregåendeAffärsmodellen som en signal om förtroendeNästa →Self-hosting som yrkespraxis

Senaste läsning

- Reflektion · 29 juni 2026 Du är inte anonym
- Reflektion · 27 maj 2026 Det en signatur inte kan fixa
- Analys · 26 maj 2026 Verklig vs skenbar integritet: Frågorna man bör ställa sig

Ta med dig den här artikeln dit du behöver den.

↓ Markdown ↓ Klartext ↓ PDF

Filen laddas ner till din enhet. Därifrån kan du spara den, importera den till Solo2 eller dela den var du vill. Cuadernos bestämmer inte destinationen åt dig.

Lacksigill · SHA-256 12cfbf4af48fb41657fbafc893a2afe63e6026314cd21fc9d22c8f38464542a8

Cuadernos Lacre · En utgåva från Menzuri Gestión S.L. ·
skriven av R.Eugenio · redigerad av teamet bakom Solo2.